Pascal
NEWSLETTER

NUMBER 9 OREGON SOFTWARE FALL 1984

Pascal-2 spans DEC line with VMS release

Oregon Software's highly optimizing Pascal–2 compiler is now available for the VMS operating system on the VAX and MicroVAX.

Release of the VMS native compiler makes Pascal–2 the only Pascal compiler that provides a uniform development environment across the full line of Digital systems from the Pro 350 to the VAX.

Unlike other language compilers available on DEC systems, all versions of Oregon Software's Pascal–2 provide identical language features, including the same interface with the operating system and hardware and the same set of language extensions. All versions contain identical programming development tools, including an interactive, high-level debugger.

Pascal–2 is integrated into the VAX/VMS system. The compiler allows calls to separately compiled routines written in Pascal, FORTRAN, or MACRO, giving developers access to existing software libraries.

In addition to DEC systems, Pascal–2 runs on M68000-based computers under the UNIX and VERSAdos operating systems.

For current Pascal–2 users, this means that applications code may be moved quickly and easily among many popular operating systems and that development work among these systems may be done with a uniform set of tools.

Tools aid developers

Like Oregon Software's other compiler systems, Pascal–2 for VMS is designed to minimize the time users spend correcting errors and developing products.

Pascal–2/VMS includes the standard kit of software tools provided with all Pascal–2 systems. These include the interactive debugger, program and text formatters; cross-referencers for program identifiers and procedures; a dynamic string package written in standard Pascal; and a set of definitions to allow Pascal-like use of MACRO code.

Compile-time error checking ensures conformance of data types to the ISO standard, locates many uninitialized variables, and detects nearly 150 Pascal syntax errors. Comprehensive run-time checking detects array index errors, illegal subrange assignments, nil pointer references, nonexistent case labels, and I/O and arithmetic errors.

In addition to generating error messages for run-time errors, the compiler produces a trace, or walkback, of the source statements leading to the error. Users are able to recover from normally fatal I/O errors and even to write their own I/O error-recovery code. Future releases of the Pascal–2/VMS compiler will provide the execution profiler and the capability to customize error messages for fatal run-time errors, which are already included in our other Pascal–2 systems. Run-time checking may be disabled for maximum performance.

Pascal–2 supports standard, optimizes code

Pascal–2 for the VAX supports all features of standard Pascal, including packed data structures and set types of up to 256 elements. Pascal–2 supports integer types of 1 to 32 bits. Pascal–2 adheres to Level 1 of the international standard (ISO 7185), which includes conformant array parameters to provide dynamic arrays.

The Pascal–2 compiler performs eight types of code optimization designed to generate fast, small code: global register allocation, common subexpression elimination, expression targeting, array index simplification, range tracking, constant folding, dead code elimination, and short-circuit evaluation.

A compilation switch disables the compiler's extended language features, allowing only ISO standard Pascal programs to be compiled. The "standard" switch generates error messages giving the location of all non-standard code, simplifying the conversion of non-standard Pascal programs to Pascal–2.

In this issue...

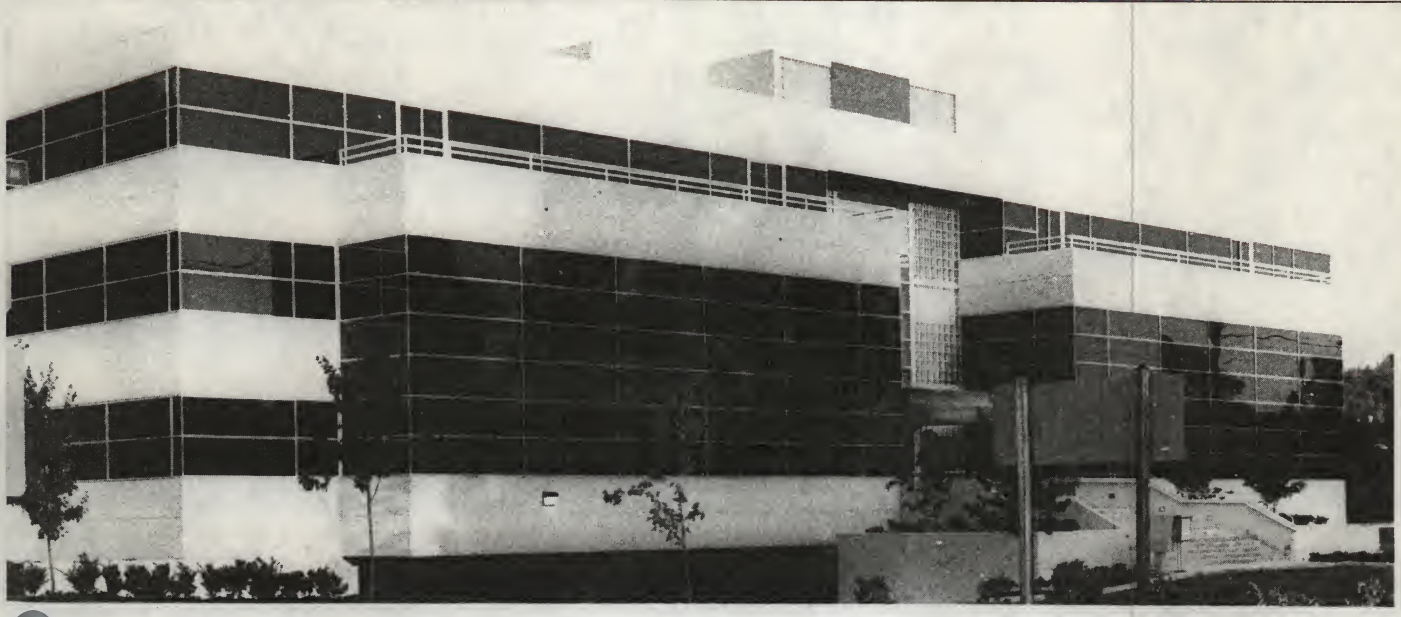


photo by David Spencer

New location – The entire second floor is devoted to staff offices, meeting rooms, a technical library, and the computer room. The employees' cafeteria, a large meeting room, and a large storage room occupy about a third of the bottom floor.

Oregon Software begins new phase

A lot has happened at Oregon Software in the past few months. Perhaps the most exciting event for our staff has been our move to new offices on September 10th. We enjoyed our old building on Canyon Road for its casual atmosphere and for its location, near downtown and next to one of the largest city parks in the country. We did not enjoy the cramped conditions with 50 people crammed into our three separate buildings with a total of 9,000 square feet. Our new location is still near downtown, across the street from a very nice riverfront park, and the building gives us 15,500 square feet with plenty of room to grow.

We passed a major milestone in August when we placed Pascal–2 for VAX/VMS in field test. We are pleased to announce that this product will be available for delivery as of October 22, 1984. Most PDP-11 or M68000 Pascal–2 programs can now be recompiled to run in VAX native mode with little or no change.

We've heard from a number of our best customers that our product support has become inadequate. This news is very disturbing, and we are taking some definite steps to solve the problem. There are two areas of concern: bug-fixing and phone support.

We are working very hard to catch up on the backlog of trouble reports that has built up over the past year, especially for our cross-development software from the VAX to the Motorola M68000. We have hired some new programmers with compiler experience who are working hard to learn Pascal–2 internals and to get problems fixed. The "old hands" on staff are giving them lots of help. For all products, we will provide maintenance releases twice a year, on a set schedule, as we do for our PDP-11 products.

We are also concentrating on teaching our phone support people more about our products and the programming environments in which they are used. Many of these staff members are relatively new to Oregon Software, and they are eager to learn more of the programming folklore that goes with our products.

We are confident that these steps will improve the quality of our technical support and we are grateful to our customers for their feedback on the level of service we provide.

Don R Baccus

Don Baccus, President

Come see us at Booth 2836

Oregon Software will be exhibiting products at DEXPO West, held at the Disneyland Hotel in Anaheim, California, December 11 - 14. Our sales staff will demonstrate new products, including Pascal–2 for VAX/VMS. Technical staff will answer your questions about existing products. We invite all our customers to visit us.

Information exchange

If you need information on technical applications involving Pascal, or if you have an application that might interest other users, send us a brief description for inclusion in the Information Exchange. Your description should follow the format of the items below. Interested parties can contact one another directly.

Two software tools help programmers write correct, portable Pascal programs and help compiler writers produce accurate, bug-free, fully conformant standard Pascal compilers. The Pascal Validation Suite Quality Control Package (PVS) consists of 734 test programs which systematically exercise a Pascal compiler to determine its ability to process programs written in ISO standard Pascal. The Standard Pascal Model Implementation (SPMI) consists of a compiler and interpreter, used in combination to process the validation suite tests correctly, and a static checker which audits Pascal programs for conformity to the ISO standard. The entire SPMI implementation is written in Pascal. The PVS and SPMI are available in machine readable source code on tape and diskette. For more information, contact: Donna Kish, Software Consulting Services, Ben Franklin Technology Center 125, Murray H. Goodman Campus, Lehigh University, Bethlehem PA 18015, (215) 861-7920.

PRM-11 is a Pascal Record Management system for use with Pascal–1 or Pascal–2 under RSX-11M V4.0. PRM-11 also runs under VAX/VMS V3.1 in compatibility mode, with the restriction that shared files may not be opened with `write` access. Documentation is included. Contact: Doug Bliss, Toledo Scale, 1150 Dearborn Drive, Columbus OH 43229, (614) 438-4877.

OPUS Communiqué

Oregon Pascal Users Society

The column for this issue consists of the OPUS membership list for September 1984. We've arranged the pages at the back of the newsletter so that you can cut them along the inside edge, staple them in the middle, and make a 5.5 by 8 inch booklet. Further information will be in our winter issue.

Who to call...

To be connected with the "right" person when you call Oregon Software, you can ask for one of the following people by name or tell the receptionist the type of information you want.

Technical matters – tell the receptionist that you need technical help. She will connect you with the support person for that day.

Updates or support status – ask for Customer Service Manager, Claire Lematta, or explain to the receptionist that you are calling about support renewal.

Manual orders – direct your request to Diane Likens.

Purchases and product information – ask for Sales or tell the receptionist what information you wish. Sales representatives Tim McMenamin, Penny Green, Dan Kane, and Lorie Griffith serve prospects and customers within the United States.

International distributors – For sales inquiries from outside the U.S., ask to speak to Linda Lausman.

Domestic distributors and OEM accounts – ask for Mary Erichsen, our Sales Manager.

Pascal NEWSLETTER
OREGON SOFTWARE

6915 SW Macadam Avenue
Portland, Oregon 97219

Editor David Spencer
Staff Writers
Thomas E. Hanrahan, Mary Hutton, Louise Waitt
Production Assistants
Betsy Slonaker, Jennifer Mulder

The Pascal Newsletter is published quarterly by Oregon Software, Inc., 6915 SW Macadam Avenue, Portland, OR 97219; (503) 245-2202. Each customer of Oregon Software receives one free subscription per site. Additional subscriptions are available upon written request.

Pascal tasks share memory under RSX

by Steve Poulsen

Even if your computer has a megabyte of memory, and even if your DEC sales representative says otherwise, no PDP-11 program accesses more than 64K bytes of memory at a time. The limitation is inherent in the PDP-11 architecture: Since addresses are 16 bits long, a single program can access only 65536 (64K) bytes or 32768 (32K) words of memory.

With large software projects, this may be a major inconvenience, because programmers must split up any large program into several smaller parts. The use of active page registers (APRs) to share memory between Pascal tasks on the RSX operating system allows these smaller parts to communicate.

First, some background. PDP-11 hardware registers are 16 bits wide, and instruction lengths are multiples of 16 bits. More important, memory addresses are stored as 16-bit values. On most PDP-11 computers, the memory management hardware is used to map virtual addresses to physical memory addresses. Virtual addresses, which are the 16-bit addresses used by a single program, are always in the range of 0 to 65535. For each virtual address, the memory management hardware computes an 18-bit or 22-bit physical address. Physical addresses may range from 0 to several million. The allocation of physical memory to tasks is a primary responsibility of the operating system.

You can use the APR in one task to map to the same physical addresses as an APR in another task, which allows the two tasks to share memory (data). This requires four steps, described in detail below: 1) create a special portion of physical memory called a partition to hold the data to be shared; 2) create a MACRO file that allows the Task Builder to map tasks to that partition; 3) install the task image for the partition; 4) use `loophole` to force Pascal variables to the start of the partition.

Before creating a partition, you must understand the allocation of virtual memory within a single task, which is controlled by 8 APRs. Each APR controls up to 8K bytes of memory. The following table shows the address ranges controlled by each APR.

The memory management hardware uses the upper three bits of the 16-bit virtual address to determine which APR (0 to 7) to use to map the physical address. The remaining 13 bits specify an offset in the range from 0 to 8191 bytes from the physical address mapped by the APR. Each APR has a length and an access mode associated with it. The length specifies the number of 64-byte blocks mapped by the register, and the access mode describes the kind of references that can be made (read-write, read-only, or no access).

Address Ranges Controlled by APRs

Octal Addresses	APR
000000-017777	0
020000-037777	1
040000-057777	2
060000-077777	3
100000-117777	4
120000-137777	5
140000-157777	6
160000-177777	7

Consider a program containing 30000 (octal) bytes or 6K words. For this task, the operating system sets APR 0 to map virtual addresses from 0 to 017777, its limit, and APR 1 is set to map addresses from 020000 to 027777, to make a total of 30000 bytes. Both APRs are set with read-write access automatically by the system. APRs 2 through 7 would be set to "no access." If the program attempts to access a virtual address in the range 040000 to 177777, a "memory protection violation" is generated. The allocation of virtual memory can be controlled by the Task Builder or by special system services called at run-time. A Pascal program allocates the heap and stack by asking the operating system to allocate additional virtual memory to the task as needed.

As an example, assume that you want to share 80 bytes of data between two Pascal programs. In particular, you want to write one program to leave a message in the shared memory area and a second one to retrieve and print the message.

You need to create a partition to hold the data. The partition specifies which area of physical memory is to be shared and prevents the operating system from using the memory for other purposes. Your system manager can create the partition dynamically or with VMR, so that the partition is installed each time the system is booted. In either case the commands are similar.

Your system is probably already making full use of available physical memory. To add a new partition, you need to reduce the size of an existing partition. This can be done with the SET/TOP command. The GEN partition is usually the largest partition, and it can usually be reduced in size without adverse side-effects. Use the PAR command to view the current partition allocation.

You want to create a common partition. Note that the GEN partition in Figure 1-A holds 515700 bytes and occupies physical memory in the range of 242100 to 760000 (760000 = 242100 + 515700). You want to allocate 80 (decimal) bytes of data. Since partitions are always multiples of 64 bytes, round the number needed up to 128 bytes (or 200 bytes in octal). To make room for 200 bytes at the end of the GEN partition use the command:

```
>SET /TOP=GEN:-2
```

The -2 parameter means that the top of the GEN partition is reduced by 200 bytes. (Memory addresses and sizes are in multiples of 100 octal bytes, so the final two zeroes are dropped.) By reducing the size of the GEN partition, you leave room for your common area DATA in the range 757600 to 760000 (757600 = 760000 - 200). To create the DATA common area, use the command:

```
>SET /MAIN=DATA:7576:2:COM
```

This creates a common partition called DATA based at 757600 with a length of 200 (octal) bytes. Use the PAR command to see the new partition (see Figure 1-B).

If you use VMR to create the DATA partition, reboot your system to make the partition available.

MACRO file describes data

After creating the DATA partition, you must next create a MACRO file to specify the amount of data to be placed in the partition. This enables the Task Builder to map tasks to the partition. (The size of the data need not be the same size as the partition, though the partition must always be at least as large as the data. This is why both the partition and the MACRO program are needed. In the next example, the partition size is the same size as the data, but you could—wastefully—put a small amount of data in a large partition.)

To share the 80 bytes of data, create a MACRO source file called DATA.MAC that allocates 80 bytes of data. The .BLKB directive allocates 80 bytes of data.

```
.TITLE  DATA
.BLKB 80.

.END
```

The decimal point after the 80 is required to interpret the number as a decimal number rather than an octal number. This file is assembled to produce DATA.OBJ.

```
>MAC DATA=DATA
```

```
>PAR
EXCOM1 067734 070000 014700 MAIN COM
EXCOM2 067670 104700 010200 MAIN COM
LDRPAR 067624 115100 002600 MAIN TASK
TTPAR  067260 117700 030000 MAIN TASK
DRVPAR 066734 147700 003700 MAIN SYS
       066670 147700 002300 SUB  DRIVER -DL:
       066570 152200 001400 SUB  DRIVER -DX:
SYSPAR 066470 153600 010100 MAIN TASK
FCSRES 066424 163700 032000 MAIN COM
FCPPAR 066360 215700 024200 MAIN SYS
       041204 215700 024200 SUB  (F11ACP)
GEN    066314 242100 515700 MAIN SYS
       041600 242100 020000 SUB  (...MCR)
```

A. Existing Partition

```
>PAR
EXCOM1 067734 070000 014700 MAIN COM
EXCOM2 067670 104700 010200 MAIN COM
LDRPAR 067624 115100 002600 MAIN TASK
TTPAR  067260 117700 030000 MAIN TASK
DRVPAR 066734 147700 003700 MAIN SYS
       066670 147700 002300 SUB  DRIVER -DL:
       066570 152200 001400 SUB  DRIVER -DX:
SYSPAR 066470 153600 010100 MAIN TASK
FCSRES 066424 163700 032000 MAIN COM
FCPPAR 066360 215700 024200 MAIN SYS
       041204 215700 024200 SUB  (F11ACP)
GEN    066314 242100 515500 MAIN SYS
       041600 242100 020000 SUB  (...MCR)
DATA   041430 757600 000200 MAIN COM
```

B. New Partition

Figure 1. Partition Allocation

The first column is the name of the partition. The second column is the internal location of the partition control block within the monitor and is not too useful. The third column is the base address of each partition or subpartition. The fourth column is the length of the partition. The fifth column is the type of the partition. MAIN is the main partition while SUB is a subpartition of a MAIN partition. The sixth column shows that a partition can be a common area (COM), a system-controlled partition (SYS), a task-controlled partition, a device driver (DRIVER), or a device common (DEV).

Installing the task image

The Task Builder creates the task image and symbol table for the partition. Several special options must be used. The **/-HD** option creates a task image without a task header, and the **STACK=0** option prevents the allocation of stack space in the task. The PAR option gives the name of the partition in which the task will be installed (DATA), the virtual base address of the partition (160000, the base address for APR 7) and the size of the partition (rounded up to 200 bytes). All numbers are expressed in octal. APR 7 is used to map the shared data because it is the highest available APR (see table). Since Pascal tasks may use additional APRs for the heap as they expand, it is best to use APRs at the high end of virtual memory. This provides the maximum memory area for the program.

You must use Task Builder commands to create three files: DATA.TSK, the task image for the partition; DATA.MAP, the map file; and DATA.SYM, the symbol table, which describes the partition and its contents. The symbol table is used by the Task Builder when Pascal programs are linked with the shared data area, as explained further below.

```
>TKB
TKB>DATA/-HD,DATA,DATA=DATA
TKB>/
ENTER OPTIONS:
TKB>STACK=0
TKB>PAR=DATA:160000:200
TKB>//
```

DATA.TSK should be installed with the INS command:

```
>INS DATA
```

'Loophole' forces data to virtual memory

You can now write Pascal programs that share the data. To specify which data is to be shared, you must force the Pascal variables to appear at virtual location 160000 (octal). Any data at that virtual location is mapped with APR 7. Then use a special Task Builder option to cause APR 7 to map to the DATA partition you created above.

Using Pascal–2, you can force data to be referenced at a particular virtual address by creating a pointer to the data. Then assign the pointer a virtual address with Pascal–2's **loophole** function.

Sample program WRITE shows how to force variables to appear at virtual location 16000 with the **loophole** function. The **type** statement does not allocate any memory for the **message** you want to share. It only describes what a message looks like. The **type message_pointer** is a pointer to a message. The only data allocated to the program is **p** which is a pointer to a message. In the first statement of the program, the **loophole** function converts the octal integer 160000 to a **message_pointer** which can then be assigned to the pointer **p**. The second statement of the program prompts you to type a message and the third statement reads the message. Since the variable **p** is a pointer to a **message**, **p^** is the message itself. Note that the storage for the **message** is not contained in the program itself, but is allocated in the small MACRO program installed in the DATA partition.

```
program WriteMessage;
  type
    message = packed array [1..80] of char;
    message_pointer = ^message;
  var
    p: message_pointer;
begin
  p:=loophole(message_pointer,160000B);
  write('Type a message:');
      readln(p^);
end.
```

Compile the program WRITE.PAS:

```
>PAS WRITE
>TKB
TKB>WRITE/FP/CP,WRITE=WRITE,LB:[1,1]PASLIB/LB
TKB>/
ENTER OPTIONS:
TKB>RESCOM=DATA/RW
TKB>//
```

The RESCOM option specifies the location of the symbol table file, DATA.SYM, which describes the shared data area DATA. The /RW switch requests both read and write access to the common data area. The /RO switch can be used to request read-only access. If you copy DATA.TSK and DATA.SYM to LB:[1,1], then instead of RESCOM, use the COMMON option: **COMMON=DATA:RW**.

You can write a similar program to read the message stored in the common data area.

```
program ReadMessage;
  type
    message = packed array [1..80] of char;
    message_pointer = ^message;
  var
    p: message_pointer;
begin
  p:=loophole(message_pointer,160000B);
  writeln('Message: ',p^);
end.
```

Here, the pointer p is initialized as in the first program. The second statement prints out the message pointed to by p. Compile and build this program (READ.PAS) as follows:

```
>PAS READ
>TKB
TKB>READ/FP/CP,READ=READ,LB:[1,1]PASLIB/LB
TKB>/
ENTER OPTIONS:
TKB>RESCOM=DATA/RO
TKB>//
```

The RESCOM option causes the Task Builder to read the file DATA.SYM to obtain information about the shared common area called DATA.

Now you can use the WRITE program to write a message to the shared data area, and the READ program to read back the information.

```
>RUN WRITE
Type a message: this is a test

>RUN READ
Message: this is a test

>RUN WRITE
Type a message: so is this

>RUN READ
Message: so is this

>RUN READ
Message: so is this
```

Setting up a partition for the sharing of data and using `loophole` to reference it allows individual programs to access more than 32K words of *total* data, even though no more than 32K words of data may be accessed at any one instant. In practical terms, the maximum additional amount that can be shared is 24K words, because even the smallest Pascal program on RSX takes up about 8K words of memory.

New staff adds to Oregon Software mix

Assistant Systems Manager, **Jim Anderson** joins Oregon Software after 17 years with Boeing Computer Services. In addition to his job, which Jim describes as "beautiful," he enjoys photography, playing guitar and banjo, writing poetry and working on hydroplanes and drag racers with his brother.

David Barnes joined the Technical Publications group this September. A graduate of Ohio State in communications theory, Dave has been a technical writer at a large company in Columbus for five years. He enjoys learning computer languages and has experience writing programs in Macro-10, BASIC, FORTRAN, PL/1, and BLISS. Dave claims "only a reading knowledge of Pascal and C" and looks forward to Pascal programming at Oregon Software. Living in the Pacific Northwest is a new experience for the Barnes family: "I'll have to learn to grow roses," Dave says.

Accounting Clerk, **Julie Berrigan** performs many tasks in our accounting department, especially processing and collecting accounts receivable. Her one-year-old, Matthew, keeps Julie running when she isn't at Oregon Software.

JoAnn Bertram became Department Secretary for Sales and Marketing in July. She claims to be the only University of Washington graduate who took ten years to get a BA in English literature; she was also putting her husband through school and taking care of two children at the time. Currently, JoAnn and her husband are remodeling their house.

photo by David Spencer

Voices of Oregon Software – Katie Wilding (left) has trained Rita Gorham (center) to take her place as receptionist. Sharon Lodewick (seated) serves as back-up on the phones.

Denise Bristow is improving customer support by organizing all trouble reports by product. She received a degree from Portland Community College in application programming last March and joined Oregon Software in August. Denise has travelled extensively in Europe, North Africa and Hawaii. She likes jogging, rafting, hiking, mountain climbing and scuba diving. She hopes to take up racquetball again now that she's out of school.

Continued on Page 10

Errors, additions to manuals

The Technical Writing staff keeps a list of changes and additions to be included in each manual. We receive information from readers through Documentation Evaluation Reports (the DER forms at the back of each manual), Trouble Reports, and support calls. Every six months, usually coinciding with a release of the software, we issue an update package.

The cycle of update packages is staggered. During the interim between updates, we list changes and additions in this newsletter and in the Release Notes.

All Pascal-2 manuals

We've found the following errors in *Pascal–2 User Manuals,* Version 2.1 for RSX, RSTS/E, RT-11, and UNIX.

ug in a Debugger example

The description of the use of negative numbers as *count* parameters for the L command is incorrect. A negative *count* parameter causes the Debugger to list statements up to and including the statement number indicated in the command. The example should read:

```
} L(Rotate,4,-2)
   16    3         A[I] := A[I+1];
   17    4       A[Last] := A[First];
```

Language Specification, extended-range arithmetic

Substitute the following passage for the existing third and fourth sentences of the paragraph:

> Normal arithmetic operations, with the exception of division and modulo, are performed on extended-range variables. Comparisons and division are signed.

Pascal-2 RSX manual

The *Pascal–2 User Manual, V2.1 for RSX,* contains the following errors.

Overlays, page 2-19

In the TKB multiline command example, delete the third line; the /MP switch eliminates the need for this entry.

Support library, page 2-21

Under "Support Library Data Definitions," the words "The file" (fourth line, first paragraph) should say "LIBDEF" to clarify which file defines the file name block.

Multiple source files, page 2-53

In the source file EXAMPL.PAS the first **%include** directive (**%include 'config'**) should be deleted so the command at the bottom of the page does not include the file twice.

Location of the compiler's work files, page 2-54

In the fourth paragraph, the "**time**" compilation switch should be "**times**," for completeness.

Using event flags, page 2-67

The example program DELAY fails to take into account that an enumerated type with fewer than 256 members is allocated one byte. Potential problems arise when the enumerated parameter is passed to the **nonpascal** procedure MARK because MARK actually requires that a two-byte (word) parameter be passed. Defining the individual time intervals to be constant integers takes care of the problem because the size allocation for integers is two bytes. In the example, change:

```
type
  TimeInterval = (Ticks, Seconds,
                  Minutes, Hours)
```

to read:

```
const
  Ticks = 1;
  Seconds = 2;
  Minutes = 3;
  Hours = 4;
```

Margins, page 5-48

In the third paragraph, the default value of the left margin (L10) should be L0. The value in the table immediately preceding that paragraph is correct.

Pascal Procedures in Resident Library, pages 2-77,78

On each page, the number 5353 octal should be 5354.

Existing Procedures in a Resident Library, page 2-79

The first full line of the task build command at the top of the page should read:

```
TKB>WRTSUM/FP/CP,WRTSUM=WRTSUM,LB:
    [1,1]PASLIB/LB:$FCSJT,LB:[1,1]PASLIB/LB
```

'Origin' Declaration, page 3-20

The reference to the section "Size Function" should be referencing the section "Size and Bitsize Function."

Debugger Guide, pages 4-4 to 4-19

When beginning a debugging session, the 2.1B Debugger identifies the program being debugged in a slightly different

way than the 2.1A Debugger. Examples on these pages should show the corrected line: **Debugging program ROTAT.**

Example, page 5-34

In the **2$:** block of the MACRO-11 example, the variable **r0** in the comment at the end of the **bne** statement should be **n**.

In the last paragraph, the word **restore** should be **rsave**.

Pascal-2 RSTS/E manual

The *Pascal–2 User Manual, V2.1 for RSTS/E*, contains the following errors.

Implementation of escape differs for RSTS

RSTS interprets the standard **escape** character **chr(27)** to be a line-terminator and returns a **$** prompt. To get a true escape character, you must set the high-order bit by using **chr(155)**.

Example: function 'NewOK', page 2-35

In the listing of **NewOK**, delete the reference to **$$HEAP** in the comment for the **space** function definition.

Error termination Status, page 2-50

In the second sentence of the note at the end of this section, the words "It is" should be inserted before the word "provided."

Pascal-2 RT-11

The *Pascal–2 User Manual, Version 2.1 for RT-11*, contains the following errors.

Copyright, page ii

The trademark entries should include "TSX-Plus is a trademark of S &H Computer Systems, Inc."

Example: function 'NewOK', page 2-30

In the listing of **NewOK**, delete the reference to **$$HEAP** in the comment for the **space** function definition.

Multiple source files, page 2-44

In the fifth paragraph, first sentence, the second "are" should be deleted.

EXITST walkback for 'severe error' status (4)

The **Exitst** procedure is a support library routine that sets the termination status of a program and stops the program when a "severe error" status is detected. The procedure's integer argument determines the termination status for any program that calls it. When a "severe error" status of 4 is passed, the procedure also invokes the post-mortem analyzer to create a walkback of the program execution from the point of failure.

Pascal-2 UNIX manual

The *Pascal–2 User Manual, V2.1 for UNIX*, contains this error:

Storage allocation for packed record, page 2-17.

For 68000 users, the last line should read "beginning at bit 8 (most significant bit)."

All Pascal-1 manuals

With the release of V1.3D, we'll issue a new user manual. In the meantime, the V1.2K manual and a supplement are the documentation.

Run-time checking switch is renamed

Oregon Software provides a set of notes entitled *Conversion From Pascal–1 to Pascal–2* for those customers upgrading from Pascal–1 V1.2 to Pascal–2 V2.1. If you have this document, please make this change. On page 1-7, you are told to change the embedded directive **$A** to **$arraycheck**. The correct change is to **$indexcheck**.

Addition to supplement

Users should note that the *Supplement to Existing Manuals* for Version 1.3 does not describe a newly added feature: Pascal–1 Version 1.3 now accepts **$** in identifiers.

Pascal-2 VERSAdos V2.0 manual

The following items should be added to *Pascal–2 Version 2.0/M68000 User Manual.*

Debugger, page 126

The description of the use of negative numbers as *count* parameters for the **L** command is incorrect. A negative *count* parameter causes the Debugger to list statements up to and including the statement number indicated in the command. The example should read:

```
} L(Rotate,4,-2)
   16    3          A[I] := A[I+1];
   17    4       A[Last] := A[First];
```

Error termination status, page 37

The following text should be added to the section on "Run-Time Error Reporting."

Both the Pascal–2 compiler and Pascal programs return a termination status when they exit. The Pascal–2 compiler terminates with a "severe error" status if it detects

compilation errors. Upon detecting an error while running, such as "subscript out of bounds," a Pascal program also terminates with a "severe error" status. Otherwise, a "successful completion" status is returned.

The Pascal support library contains a routine that may be called from Pascal programs to set the termination status and stop the program. To use this feature, declare an external procedure named **exitst**. This procedure, defined in the support library, takes an integer argument, as shown:

```
procedure Exitst(Status: integer);
{ procedure declaration }
  external;
```

Call the procedure at a point in the program where you want to exit in case of a severe error, as shown:

```
begin       { program Severe }
  .
  .
  .
Exitst(4);    terminate with severe status
  .
  .
  .
end.        { program Severe }
```

A status of 1 means normal termination; any other status means that an error terminated the program.

Compilation Switch /longlib

In order to take advantage of "short references and instructions" which use 16-bit addresses, the Pascal–2 support library under VERSAdos resides in the low end (first 32K bytes) of memory. Support library subroutines use short references internally and the compiler generates calls to the support library using the 16-bit addresses.

The new **/longlib** switch causes the compiler to generate full 32-bit address calls to the support library. Users who wish to use **/longlib** must change all internal use of short instructions in the library's source code before relocating the support library.

Concurrent Programming manual

Users of V1.0A should make this change.

Booting Instructions, page 4-10

You are told to use the supplied program MAKEBOOT to convert the load module to a bootable image. Users must also compile and assemble FHSCAL.SA and FHSUTI.PA, two modules that are shipped with the utilities. These modules are then linked with MAKEBOOT.

New staff

From Page 7

Lyn Gabel, Administrative Assistant to our President, has been writing a company personnel manual and will be doing research projects. A graduate of Oral Roberts University with a BA in psychology, Lyn says "I was a Marine brat, so we lived all over the U.S. when I was growing up." Lyn and her husband are Portland Winter Hawks boosters; they board one of the team's defensive players, a student from Canada, during the hockey season. Fortunately, Lyn "loves to cook."

Rita Gorham came from 4A's Temporary Service to fill in as our receptionist. She did such a good job that we asked her to stay. Rita enjoys cross-country bicycle touring and downhill skiing. She hopes to take up scuba diving so she can add seashells to her collection, preferably some from the Barrier Reef area of Australia.

Formerly an intern programmer, **Bruce Graham** became a full-time programmer in August. A graduate of Reed College in mathematics with additional coursework in computer science at OSU, Bruce has been working on the UNIX debugger. Before joining Oregon Software, Bruce spent five years in Alaska as a fire fighter and taught high school math and physics in New Zealand. He enjoys woodworking, particularly building boats.

Technical writer **Mary Hutton** joined Oregon Software in July and immediately began revising manuals for new releases of the software. A graduate of both Stanford (BA in Latin) and the University of Washington (BS in scientific and technical communication), her interests include archaeology, mathematics, language study, anthropology, and all types of crafts.

Sales representative **Dan Kane**, a Portland native, recently graduated from Oregon State University (BS degree in Business Administration) where he concentrated on marketing and sales management. Dan says he likes sales because it involves communicating with people. His avocations are playing as much golf as possible, playing softball, and skiing.

Krzysztof Krüger, a native of Poland, comes to Oregon Software via Aachen, Germany, where he programmed in Pascal-2 for three years. In addition to Polish and German, Kris is fluent in Russian, French, and English. After he gets his family settled, he hopes to resume his favorite hobbies, basketball and photography. What does he think of his new home? "I'm glad to be here," says Kris.

Continued on Page 15

The Log: errors, work-arounds, changes

As in previous issues, this log also describes the significant changes in Pascal-1, Pascal-2, and SourceTools. As a service to users who may not have the current release, we supply work-arounds where possible.

For the first time, we've listed many known bugs: those problems reported by users and verified as bugs by Oregon Software. By conveying this information, we spare you the labor of tracing and reporting bugs unnecessarily and give you some indication of a particular report's status.

To use this log, you must know the release level of your software. (The release number is printed on the headline of all program listings.) Then review the log to determine the changes made since the release of your version. As an additional reference, we've placed the Trouble Report numbers in square brackets at the end of each log entry.

If the changes are of particular importance to your application, your Designated Contact Person should request an update, in writing. Upon receipt of your written request, you will receive the latest version of the software.

Pascal-1

Version 1.3C has just been released. We have not received any Trouble Reports from users.

Pascal-2

Version 2.1D corrects a number of problems common to PDP-11s with RSTS/E, RT-11, or RSX-11 operating systems.

Packed boolean array problems

The compiler no longer generates incorrect code when a packed array of boolean crosses a byte boundary. Under V2.1C, only the lower byte of the word was fetched so bits representing the higher elements of the array were not set properly [1173, 1319, 1986].

Declared as a **var** parameter, the packed array of boolean is now passed to a procedure correctly [1269].

Record field as a parameter

The compiler no longer generates bad code when certain nested record fields are passed as a parameter [1191].

Negative stack offset for common subexpressions

Depending upon the location of a variable's declaration, two identical calculations involving certain functions, including sine and cosine, could have different results. The library routines for the functions no longer cause the compiler to reference a negative offset from the stack pointer [1572].

Wrong results from integer comparison

A comparison between an integer and a subrange that is an element of a packed record no longer produces an incorrect result [1572].

Incorrectly addressed real variable

Certain globally allocated real variables are now addressed correctly in large programs [1613].

Dynamic string package errors

The `Insert` routine no longer prints the error message `Array subscript out of range` if the combined length of the two input strings exceeds the target string's length. Instead, the insert string is concatenated onto the end of the target string and truncated [1782].

The `Concatenate` routine no longer reports errors when the string to be concatenated overflows the target string. The input string is truncated to fit.

%Page directive doesn't work

On listings, `%page` now increments the page number correctly [1914].

Reserved instruction trap error

Using a **packed record** containing integers in a nested procedure call no longer causes a reserved instruction trap [1988].

Utilities don't compile

With Versions 2.1A and 2.1B, attempts to compile the utilities resulted in an `Unknown Pascal run-time error` [2027, 2303a, 2393].

Illegal instruction gives "compiler writer error"

Using certain illegal instructions resulted in the error message for internal problems instead of the appropriate error message [2042].

Sets generate bad code

Set constructor expressions of the form **[var1..var2]** no longer generate bad code at times [2046, 2046a].

No error message

The compiler now gives an error message for attempts to pass an element of a **packed** structure as a variable-parameter [2151].

Incorrect use of 'R0' for procedure calls

The compiler now saves the contents of the `R0` register before a call to a **nonpascal** procedure [2177a].

Set type produces wrong results

The declaration and statement shown below now compile correctly [2177b].

```
type  a = set of (one,two,three);
var   b:a;

begin
  b:= [one,three];
  write(b * [] <= []);
end.
```

Function returns incorrect result

The compiler now correctly changes bit lengths to byte lengths when a function returns a structured type [2177c].

Bit optimizations incorrect

Compiler optimization of certain user-defined procedure calls no longer produces an incorrect sequence of execution [2184].

PB formatter rejects 'nonpascal'

The formatter now recognizes the **nonpascal** directive and treats it exactly like the **external** directive [2202].

String comparison range wrong

For string comparisons, **ord(char)** is now in the range `0..255` instead of `-128..127` [2248].

Unsigned characters treated as signed

The compiler no longer treats 8-bit characters as if they are signed numbers at times [2329].

Assignment statements cause failure

A complicated series of assignment statements involving arrays of type **real** no longer fails. [2403].

Compiler consistency checks reported

Version 2.1D fixed certain conditions causing the error message `Undeleted temps in procedure...` [1142].

Compatibility mode address trap

The Task Builder does not give a start address to programs compiled without a main program and with the embedded **nomain** switch. When executed, such programs trap because no program may start at location 0 on the PDP-11. No error message is issued by the Task Builder or by the Pascal–2 compiler, but the code generated is correct [1989].

Changes to RT-11

Misleading error message for nonexistent files

The compiler no longer prints the error message **Unknown Pascal run-time error...** for an attempt to compile a nonexistent file or a file containing a **%include** of a nonexistent file [2235].

Changes to RSX

/apd switch problems

Programs appending records to an existing file do not occasionally crash after crossing a block boundary. Formerly, these programs trapped out to RSX with **T-bit Trap** or **BPT executions** error message and changes in the size of the program caused a different error message [2392].

'Forini' incompatibility

The FORTRAN initialization routine, **forini**, now works properly on VAX in RSX compatibility mode [2370]. (This bug was not fixed in the V2.1C release, as reported in the last issue.)

Memory protection violation

The value of `R0` is now initialized before adding the offset [1273].

Changes to UNIX

Version 2.1E is the current release of the UNIX/68000 compiler. We've corrected a number of problems in the V2.1D release and completed development work on the Debugger.

Debugger

The Pascal–2 Debugger now functions under UNIX.

Unnecessary checking in 'for' statement

The range-tracking optimization no longer performs unnecessary final-limit checks on generated code.

Bad integer error message

Certain cases of incorrect code generated by **packed** fields in records have been corrected.

Trap on exponent underflow

Support library routines now return a signed zero instead of causing a fatal run-time trap.

Floating point division

The entire divisor is now restored to the partial remainder, not just the low-order word of the divisor.

Remainders and quotients inaccurate

The correct register is now accessed for division of operands longer than 16 bits. Previously, quotients could be in error by one bit and remainders (**mod**) could suffer inexpilicable loss of significance.

Inaccurate results of 'If' conditional

The expression '`if x mod y = 0`' no longer produces inaccurate responses.

'For' loop failures

For loops with a control variable that is an **enumerated** type of one byte in size now generate correct code [1918].

Integer constant range checking

Expressions of the form *integer-constant* `in [0..255]` no longer fail.

Conformant array parameters

Conformant array parameters in **forward** and **external** procedure declarations no longer take 25 minutes to compile [2311].

Loss of stack pointer

Very large programs no longer crash due to loss of the stack pointer.

Real number accuracy

Nested functions that return a **real** value no longer generate incorrect code occasionally.

'Undeleted temps' error message

Certain **for** loops no longer generate the error message **Undeleted temps in procedure** ... [2020].

Order of declarations

Declaring an **external** function before a global **var** declaration no longer generates an error message.

Bad 'case' statement error message

An untested value within a **case** statement now returns the error message **No case provided for this value** [2469].

Changes to VERSAdos

Version 2.0N is the current release of the VERSAdos native compiler. We've corrected some of problems that were in the initial release (V2.0M) and have V2.0P, containing more bug fixes, in field test. We've skipped an O-release because V2.0O is visually and verbally awkward.

Debugger

The Pascal–2 Debugger now runs under VERSAdos.

External use of conformant array parameters

The compiler no longer rejects external use of conformant array parameters.

File access problem

The compiler can now access files in directories other than the current one.

Origin extension

Variables used with the **origin** extension now generate correct code.

Changes to cross-development systems

The Cross-Development System's initial release for RSX-hosted systems was Version 2.0M in early 1983. Version 2.0M for VAX/VMS hosts was released later in the same year.

Version 2.0N corrected the following problems for cross-development systems hosted on both VMS and RSX.

Conformant array index variable

The compiler no longer generates bad code when passing a conformant array to a subroutine as a **var** parameter and attempting to access an element of the conformant array with an index variable that was not declared at the same nesting level as the conformant array parameter. Formerly, the wrong level of addressing was used during array index calculation; enabling checking had no effect on the error.

External use of conformant array parameters

The compiler no longer rejects external use of conformant array parameters.

'For' loop with computed limits

A **for** loop with computed limits now generates correct checking code when the index variable is within range.

GLOBAL$$ constant

For modules with a main program and a global data area greater than 32K bytes, the constant now generates correct code.

/NOCHECK switch

Checking can now be turned off; this switch also disables stack overflow checking.

ORD operator

When applied to a function of type **char**, the **ord** operator now generates correct code.

Origin extension

Variables used with the **origin** extension now generate correct code.

Set types error message

The compiler now generates the appropriate error message when encountering a set constant of base type **integer** which lies outside of the 0..255 subrange limit. Previously, it trapped out to the operating system with an access violation.

Structured constant traps

A duplicate definition of a structured constant produces an appropriate error message instead of causing the compiler to trap out to the operating system with an access violation.

Changes to VMS-hosted system

Version 2.0N, for VAX/VMS systems, corrected the following problems.

Multidimensional conformant arrays

The compiler no longer traps out to VMS with a divide by zero exception when compiling a two-dimensional conformant array parameter.

Pack/Unpack

The transfer features **pack** and **unpack** now function correctly when operating on **packed array of integer** subranges that are allocated 2 bytes per element: e.g., **packed array**`[1..5]` **of** `0..1000`.

SourceTools

Version 1.0C is the current release. Newsletter #8 reported the bugs fixed in that release.

Miscellaneous notes

The Pascal Standard does not define the **mod** operation for negative divisors. If either operand of the function is negative, its result may be unexpected. Examples: -30 **mod** 8 yields a result of 2, -23 **mod** 8 yields 1, -6 **mod** 5 yields 4. Programs must check for this condition.

Known bugs

We've not fixed all the trouble reports we've received for Pascal-2. Many bugs have been logged but not verified, which means we cannot categorize or describe them by type of problem. By the next newsletter, we intend to reduce the backlog to the point where this section is more informative.

In the meantime, we're listing those trouble reports that we've verified and scheduled for subsequent releases.

Pascal-2 VERSAdos

The following problems with Version 2.0N have been reported and verified for the native compiler.

MOD function

The **mod** function does not produce an error message for negative integers. (See the explanation under "Miscellaneous Notes")

Statement numbers for input don't match output

The statement number in the assembler output code (.SA extension) does not agree with program statement numbers when files are input with the **%include** switch.

Utilities fail

The utility programs may run out of memory on larger files. Relinking them may fix the problem.

Pascal-2 cross-development systems

The following problems with the V2.0N release have been reported and verified for VMS- and RSX-hosted systems.

Bit test instruction gives illegal address

The compiler generates an illegal addressing mode when using the bit test (BTST) instruction for a construct $a\ IN\ b$ where b is a set expression evaluating to a constant at compile time and a is a variable.

Cross-assembler faulty

The OASYS Cross-Assembler occasionally rejects valid code.

Error checking code

The compiler does not generate checking code for subranges in the form `0..maxint`.

Field width error with 'writeln'

At run-time, the expression `writeln(value:n);` prints an indefinite number of blank lines under some conditions, e.g., printing hexadecimal output with the field specification `(value:-8)`.

For loop executes once only

A `for` statement in the form

`For` *integer*`:=`*integer* `div` *integer* `to` *integer* `div` *constant*

sets the loop index to the final value, so that the loop is executed once only.

Size limit for arrays

The size limit set for certain declarations of arrays is 8M bytes instead of 16M bytes.

Statement numbers for input don't match output

The statement number in the assembler output code (.SA extension) does not agree with program statement numbers when files are input with the `%include` switch.

Truncated integers

No error message is generated if the value returned when an integer read from the standard input file (`P$4`) must be truncated to fit in the storage allocated for the subrange.

Unacceptable abbreviations

PASMAT accepts only the abbreviated form `/o` for the `/options` switch.

Unacceptable file names

PASMAT does not accept long file-name arguments.

Pascal-2 RSX-hosted system

The following problems with the V2.0N release have been reported and verified.

Characters reversed

The characters of a string constant are reversed in a structured constant containing both strings and numbers.

'MOD' function

The `mod` function does not produce an error message for negative integers. (See the explanation under "Miscellaneous Notes")

New staff From Page 10

Sharon Lodewick spells Rita at our reception desk and is training to be a word processing operator. A native Oregonian, she graduated from Sunset High School last spring and will study American literature in the future, perhaps to become a writer or a teacher. She plans to go to Portland State part-time when she's not answering the Oregon Software phones.

Lois McClain became our Tele-marketing Representative in July after six years at American Data Services as a company liason to financial institutions and two years as office manager for All West Display. She plans to learn more about programming at Oregon Software. An avid football fan, Lois "never misses a game" with her six-year-old son. Like Lynn, Lois comes from a military family and has lived in many western states. She collects Oriental porcelain figurines.

Jennifer Mulder worked during the summer as an Intern in the Technical Publications group. She has gone back to BYU for another year of college, including coursework in computers.

As secretary for the Sales Department, **Julie O'Brien** helps end users, distributors, our customer service and OEM sales staff, by sending out project correspondence and literature. Julie recently graduated from Portland State with a degree in social sciences and plans to do volunteer work at the Metro Crisis Center. A native of Oregon, Julie has three children and likes to play tennis.

Cyndy Smith has joined our Technical Department as secretary. A graduate of the University of Oregon in business management, Cyndy will be "taking care of the administrative details and handling special projects" for our programmers and software engineers. Cyndy just returned from three months touring Europe and two years living in Tsuruma, Japan, where her husband worked as an architect for a military contractor. Cyndy enjoys practicing the flower arranging she studied in Japan.

Modula-2 ?

I'd like you to help us understand the growing interest in Modula-2 so that Oregon Software can better plan "if" and "how" we might approach delivering a Modula-2 product at some point in the future. This won't reduce our commitment to expanding our Pascal product lines and improving customer support.

Please telephone me at our free 800-874-8501 number or send me a photocopy of the following questionnaire with your responses plus any comments that you feel might help our planning. Ask anyone else in your organization with strong views on Modula-2 to respond as well.

1. What's your level of interest in Modula-2? Please write a short description or check the items that apply to you:

( ) a. I'm already using a Modula-2 compiler for _____

system. Provided by _____ Does it meet your needs? _____

_____

( ) b. I plan to acquire a Modula-2 compiler. When?_____

( ) c. I'm interested in Modula-2 but have no specific plans.

( ) d. I'm not particularly interested in Modula-2.

2. What capabilities not in Pascal-2 would cause you to switch to Modula-2, C, Ada, or some other language?

3. If you could get a reliable, efficient Modula-2, would it be your first choice among languages? Rank the languages you would prefer to use.

4. What type of work might you do with Modula-2? Examples: process-control, real-time, teaching, systems programming, applications, utilities, business?

5. Rank the auxiliary packages you most want to see in any new language products? Examples: debuggers, profilers, construct editors.

6. What do you think about the view of some of our customers that a new language is not needed? They suggest that we further enhance our Concurrent Programming Package (CPP) to include more of the functionality of Modula-2.

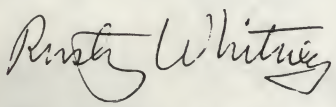
Name _____ Title/Group _____

Company/Organization _____ License Number _____

Address _____

_____ Telephone _____

Thanks in advance for helping with our product planning,



Rusty Whitney, Chairman

staple - staple

OPUS Directory

1984

Editors note: The OPUS memberhip list is not alphabetized. The original format has been retained.

If you wish, you may remove these pages from the newsletter and keep them as a separate packet. To make a 5 by 8 inch booklet, follow these directions:

1. Cut the pages along the inside edge.
2. Staple in two places on the dotted line.
3. Fold around the staples.

To put them in a three-ring binder, you must punch holes along the outer edge and cut them along the inside edge.

Mr. Jerry Shaver
Allergan
2525 Dupont Drive
Irvine,
CA 92713
UW

COMPILERS: 2.0/2.1
MACHINES: PDP 11/XX
OPERATING SYS.: RSX-11M
APPLICATIONS: Lab. Interfaces
SYSTEMS: Business

Mr. Dave Wilborn
AmicuSoft, Inc.
113 E. Savarona Way
Carson,
CA 90746
UW (213) 538-4682

COMPILERS: ALL
MACHINES: PDP 11/XX
OPERATING SYS.: V4.0
APPLICATIONS: Business
SYSTEMS: EOM, Business

Mr. Bob Caldwell
Centaurus Software Inc.
975 Hornblend, Suite B
San Diego,
CA 92109
UW (619) 270-4552

COMPILERS: 2.0/2.1
MACHINES: PDP 11/XX 11/23, 11/44, 11/24
OPERATING SYS.: RSX-11M
APPLICATIONS: Process Control, Env. Control
SYSTEMS: OEM & Custom Process Control

Mr. Gil Larson
TMI
17862 Fitch
Irvine,
CA 92714
UW

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Martin Greenberger
The Color Press
4871 West Washington Blvd.
Los Angeles,
CA 90016
UW (213) 937-9242

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Suvat Sukchindasathien
Autologic, Inc.
1050 Rancho Conejo Blvd.
Newbury Park,
CA 91320
UW (805) 498-9611 x188

COMPILERS: 2.0
MACHINES: PDP 11/XX (70)
OPERATING SYS.: RSX-11M Plus Version 2.0
APPLICATIONS: Scientific: Compiler, Linker, Library Programs Other: Fonts Programs
SYSTEMS:

Mr. Ralph Haas
1044 Calle Pecos
Thousand Oaks,
CA 91360
UW

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Ms. Christina Runtagh
Telesis
Corporation of Delaware, Inc.
21 Alpha Road
Chelmsford,
MA 01824
UE

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Jeff Hemmerling
Test Systems Strategies
7929 SW Cirrus Dr.
Bearerton
OR 97005
UE (503) 643-9281

COMPILERS: 2.0/2.1
MACHINES: PDP 11/XX 23
OPERATING SYS.: RSX-11M
APPLICATIONS: Real Time Process Control
SYSTEMS:

Mr. Donald Andruska
Senior Programmer/Analyst
Honeywell Inc.
Commercial Construction Div.
1500 West Dundee Road
MS 5600
Arlington Heights,
IL 60004
UM (312) 394-4000

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Dick Pearson
5910 Flower
Arvada,
CO 80004
UW

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Pablo Campos
8004 Willis Avenue
Panorama City,
CA 91402
UW

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Thomas Connell
Data Processing Systems Manager
Koogle & Pouls Eng., Inc.
8338A Comanche, N.E.
Albuquerque,
NM 87110
UW (505) 294-5051

COMPILERS: 2.0/2.1
MACHINES: PDP 11/XX, 11/34A 11/60
OPERATING SYS.: RSX-11M, RSTS/E
APPLICATIONS: Eng. (mapping), (graphics)
SYSTEMS: Graphics

Mr. Dave Rivard
TestMaster
3191 "D" Airport Loop Dr.
Costa Mesa,
CA 92626
UW (714) 660-0436

COMPILERS: 2.0/2.1
MACHINES: PDP 11/XX, 11/34, 11/23+ (future)
OPERATING SYS.: RSX-11M 3.2, RSX-11M/Plus (future)
APPLICATIONS: ATE
SYSTEMS: ATE

Mr. Randy Biallas
Medtronic Inc.
3055 Old Hwy. 8
Minneapolis
MN 55440
UM (612) 574-3623

COMPILERS: 2.0/2.1
MACHINES: PDP 11/XX, VAX, MC6800
OPERATING SYS.: RSX-11M
APPLICATIONS: Medical, Real-time, NC Develop.
SYSTEMS:

Mr. Don Sylwester
System Manager
Concordia College
800 North Columbia Ave.
Seward,
NB 68434
UM (402) 643-3651

COMPILERS: 2.0/2.1
MACHINES: PDP 11/XX 11/70
OPERATING SYS.: RSTS/E
APPLICATIONS: Academic
SYSTEMS: Academic

Mr. Mark Berry
Lawrence Livermore Labs
M.S. L330
P.O. Box 808
Livermore,
CA 94550
UW (415) 423-3274

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Pablo Campos
Litton Data System
8000 Woodley
Mail Drop 48-10
Van Nuys,
CA 91409
UW (213) 902-4807

COMPILERS: 1.3, 2.1
MACHINES: PDP 11/XX PDP 11/23 (LSI)
OPERATING SYS.: RT-11 04.00, RSTS/E
APPLICATIONS: Business
SYSTEMS: OEM, Busines

Mr. Ed Costello
Manager, Systems/Software
Helgeson Nuclear Services
5587 Sunol Blvd.
Pleasanton,
CA 94566
UW (415) 846-3453

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Michael Fay
Microtec
P.O. Box 60337
Sunnyvale,
CA 94088
UW (408) 733-2919

COMPILERS: 2.0/2.1
MACHINES: PDP 11/60, VAX 11/750
OPERATING SYS.: RT-11, RSX-11M
APPLICATIONS: Cross-Pascal, C for Micros
SYSTEMS:

Mr. Earl Girbovan
Proquip Inc.
1725 De La Cruz Blvd.
Building 3
Santa Clara,
CA 95050
UW (408) 496-0322

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Rick Freedman
Ross Systems, Inc.
1860 Embarcadero Rd.
Palo Alto,
CA 94061
UW (415) 856-1100

COMPILERS: 2.0/2.1
MACHINES: PDP 11/XX, VAX
OPERATING SYS.: RSTS/E
APPLICATIONS: Financial Modelling, Graphics, Data Base Mgt.
SYSTEMS:

Mr. Martin Hall
Beenhams Railway Lane
Littlemore
Oxford

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Dave Charlesworth
Systems Analyst
Department of Electronic Services
Campus Services Division
Queen's University
Kingston,
Ontario K7L 3N6
CA (613) 547-6640

COMPILERS: 1.2, 2.0
MACHINES: PDP 11/XX, 11/23
OPERATING SYS.: RT-11 4.0
APPLICATIONS: Office, Process Control, Communications
SYSTEMS:

Mr. Gerry Pelletier
Systems Engineer
Prior Data Sciences Ltd.
39 Highway 7
Ottawa,
Ontario K2H 8R2
CA (613) 820-7235

COMPILERS: 1.2/1.3, 2.0/2.1
MACHINES: PDP 11/XX, VAX, MC68000
OPERATING SYS.: RT-11, RSX-11M, RSX-11M/PLUS, RSTS/E
APPLICATIONS: Scientific, Academic, Military, Real-time Sys.
SYSTEMS: OEM, Academic, Real-time Sys.

Mr. J.D. Richardson
Defence and Civil Institute
Of Environmental Medicine
1133 Sheppard Ave. W.
P.O. Box 2000
Downsview,
Ontario
CA (416) 635-2073

COMPILERS: 1.1
MACHINES: PDP 11/XX /04,/34,/40,LSI 11/23, VAX (future)
OPERATING SYS.: RT-11 V4 and RT-11 emulator running under UNIX
APPLICATIONS: Scientific, Military
SYSTEMS: Experiment Control & Monitoring

Mr. Christopher Williams
National Research Council
Canada
Low Speed Aerodynamics
Laboratory
Bldg. M-2,Room 125,
Montreal Road
Ottawa,
Ontario K1A OR6
CA (613) 993-1141

COMPILERS: V2.1A
MACHINES: PDP-11/23, 11/34, 11/44
OPERATING SYS.: RT-11, RSX-11M
APPLICATIONS: Engineering & Scientific, real-time data acquisition & control systems for windtunnels
SYSTEMS:

Mr. Alban Cornish
Senior Systems Analyst
Canada Systems Group
1736 Courtwood Cres.
Ottawa,
Ontario K2C 2B5
CA (613) 225-1171

COMPILERS: 2.1
MACHINES: PDP 11/XX-11/23, VAX-11/780
OPERATING SYS.: RSX-11M 3.2
APPLICATIONS: Scientific: Interactive Centographic Sys., Other: Computer Vehicle Dispatch Sys.
SYSTEMS: OEM

Mr. Oddgeir Uglum
Kokums Camcar Corp.
P.O. Box 5245
Vancouver,
WA 98668
UW

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Joseph Cook
Larse Corporation
4600 Patrick Henry Dr.
Santa Clara,
CA 95050
UW (408) 988-6600 Ext 312

COMPILERS: 1.2/1.3, 2.0/2.1
MACHINES: PDP 11/XX
OPERATING SYS.: RSX-11M
APPLICATIONS: Telecommunications
SYSTEMS: OEM

Mr. John Chadwick
Sundstrand Data Control
Overlake Industrial Park
Redmond,
WA 98052
UW

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Timothy Dale
System Manager
IDD Test Engineering D/S 61-244
IDD Tektronix
P.O. Box 1000
Wilsonville,
Oregon 97070
UW

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Crawford Brewer
DCT Systems Inc.
15 Archibald Street
P.O. Box 832
Moncton, N.B. E1C 8N6
Canada
CA (506) 389-1222

COMPILERS: 1.2/1.3
MACHINES: PDP 11/XX 11/34
OPERATING SYS.: RSTS/E
APPLICATIONS: RDM Data Mgmt. Sys.
SYSTEMS: Business

Mr. Matt Richards
Research & Development
Polkaudio
1915 Annapolis Road
Baltimor,
MD 21230
UE

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. A.B. Bailey
Systems Consultant
Centre-file Limited
P.O. Box 177
75 Leman Street
London
England E1 8EX
GB 01-488 3131

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Manuel Guerra
Experiencias Industriales,
S.A.
C/O Joaquin Rodrigo
Aranjues,
Madrid
SP (91) 891 0840

COMPILERS: 2.0/2.1
MACHINES: PDP 11/XX
OPERATING SYS.: RSX-11M
APPLICATIONS: Business: Dept. Control
SYSTEMS: OEM: Real-time control, ARQ Systems

Mr. Ed Moran
Horace Mann School
231 West 246th Street
Bronx,
NY 10471
UE (212) 548-4000

COMPILERS: 2.0/2.1
MACHINES: PDP 11/XX, 11/44
OPERATING SYS.: RSTS/E
APPLICATIONS: Academic
SYSTEMS: Academic

Mr. Henry Baird
RCA Lab
Rm 201
Princeton,
NJ 08540
UE

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Mike Dearing
University of Wisconsin
Inst. Sys. Center
1500 Johnson Dr.
Madison,
WI 53706
MW (608) 263-1564

COMPILERS: 2.0/2.1
MACHINES: PDP 11/XX, VAX
OPERATING SYS.: RT-11, RSX-11M
APPLICATIONS: Scientific, Engrg.
SYSTEMS: OEM, Academic, Research

Mr. Richard Chlopan
Pros. Dir. Comp. Sci.
Westmar College
Le Mars,
IA 51031
MW (712) 546-7081 ext. 315

COMPILERS: 2.0/2.1
MACHINES: PDP 11/XX, 11/44
OPERATING SYS.: RSTS/E
APPLICATIONS: Business: Admin. Programs, Academic: Comp. Sci. Proj.
SYSTEMS: Academic: Monitors and Tools, etc.

Mr. B.J. Th. Ockeloen
System Engineer
Universiteit van Amsterdam
Vakgroep Algemene Dierkunde
(Dept. of Zoology)
Biologisch Centrum Gebouw II
Kruislaan 320
1098 SM Amsterdam,
Netherlands
NE 020) 680551 Ext 133

COMPILERS: 1.2/1.3, 2.0/2.1
MACHINES: PDP 11/XX, PDP 11/60, PDP 11/34, MC68000 (soon)
OPERATING SYS.: RSX-11M V.3.2 Autopatch E
APPLICATIONS: Scientific: Zoological, Acedemic: Pattern recognition
SYSTEMS: Academic

Mr. Sydney Davis
Computrol
15 Ethan Allen Highway
Ridgefield,
CONN 06877-6297
UE (203) 544-9371

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Dr. A. Das
Associate Professor
Computer Science
Quincy College
Quincy,
IL 62301
UE

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Prof. Allan Smith
Associate Professor
Drexel University
Department of Chemistry
Philadelphia,
PA 19104
UE (215) 895-2667

COMPILERS: 1.2
MACHINES: PDP 11/XX
OPERATING SYS.: RT-11
APPLICATIONS: Scientific: Chem. Res. Academic: Instructional Software
SYSTEMS:

Mr. Jeffrey Levine
Department of Earth and
Planetary Sciences
The Johns Hopkins University
Baltimore,
MD 21218
UE

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Robert A. Rennert
Director of Academic Computing
Kenyon College
Gambier,
OH 43022
UE (614) 427-2244 Ext. 2559

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Murray Zetterholm
Ford Motor Credit Co.
The American Rd., Room 2235
Dearborne,
MI 48121
UE

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Bob Friedman
E.I. Dupont Company
Clinical Systems Division
Glascow Site Rt 896
Glascow,
DE 19702
UE (302) 453-3280

COMPILERS: 2.0/2.1
MACHINES: PDP 11/XX, PDP 11/70, 11/44, 11/24, MC68000
OPERATING SYS.: RSX-11M, RSX-11M/PLUS
APPLICATIONS: General Sys. use Downloaders, Utilities, General Math
SYSTEMS:

Mr. Robert Lacovara
InterNet
500 Grand Avenue
Englewood,
NJ 07631
UE (201) 567-3363

COMPILERS: 1.2/1.3, 2.0/2.1
MACHINES: PDP 11/XX /23
OPERATING SYS.: RT-11,RSX-11M
APPLICATIONS: Mfg. Data Keeping
SYSTEMS: OEM, Bus.

Mr. Robert Thornton
New York Institute of
Technology
Computer Graphics Laboratory
P.O. Box 170
Old Westbury,
NY 11568
UE (516) 686-7644

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Jim Nash
JEOL USA Inc.
11 Dearborne Rd.
Peabody,
MA 01960
UE (617) 535-5900

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Alan Duberstein
Pine Instrument Co.
3345 Industrial Blvd.
Bethel Park,
PA 15102
UE (412) 831-8870

COMPILERS: 2.0/2.1
MACHINES: PDP 11/XX
OPERATING SYS.: RSX-11M, RSX-11M/PLUS Versions
APPLICATIONS:
SYSTEMS:

Mr. Hal Laurent
Psych Systems
600 Reisterstown Rd.
Baltimore,
MD 21208
UE (301) 486-2206

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Bob Schor
The Rockefeller Unviersity
1230 York Avenue
New York,
NY 10021-6399
UE

COMPILERS: 1.1, 1.2/1.3, 2.0/2.1
MACHINES: PDP 11/XX
OPERATING SYS.: RT-11 (4 & 5), TSX-11 (2 & 3)
APPLICATIONS: Scientific: Data Proc., Academic: Algorithm Testing, Teaching
SYSTEMS:

Mr. Steven Bentley
Academic Computing Center
Tougaloo College
Tougaloo,
MS 39174
UM

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Ken Tibesar
3M
3M Center, Bldg.18-1
Saint Paul,
MN 55144
UM (612) 733-8819

COMPILERS: 1.1, 1.2/1.3, 2.0/2.1
MACHINES: PDP 11/XX, VAX
OPERATING SYS.: RSX-11M, RSX-11M/PLUS, VAX-VMS
APPLICATIONS: Mfg.
SYSTEMS: Process Control

Mr. Troy Monaghen
William Rainey Harper College
Roselle and Algonquin Roads
Palatine,
IL 60067
UE (312) 397-3000

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Terry Medlin
Vice President
GEJAC Incorporated
P.O. Box 188
Riverdale,
MD 20737
UE (301) 8664-3700

COMPILERS: 2.0/2.1
MACHINES: PDP 11/XX, VAX
OPERATING SYS.: RSX-11M, VMS
APPLICATIONS: Business
SYSTEMS:

Dr. Richard J. Perry
Villanova University
Dept. of Electrical Engineering
Villanova,
PA 19085
UE (215) 645-4969

COMPILERS: 2.0/2.1
MACHINES: PDP 11/XX
OPERATING SYS.: RT-11 4.0
APPLICATIONS: Academic: Communications, Control & Signal Proc.
SYSTEMS: Academic

Mr. Harald Norvik
IKU
Organic Geochemistry Dept.
Hakon Maynussons gt. 1B
Postboks 1883
Norway
7001
NO (7) 915660

COMPILERS: 1.2/1.3
MACHINES: PDP 11/XXPDP 11/23, PDP-11/24
OPERATING SYS.: RT-11 V.4 (slightly), RSX-11M V.3.2 & 4.0 (mainly)
APPLICATIONS: Scientific: chemistry
SYSTEMS: Business

Mr. Mike McCready
Software Manager
Automation Engineering
Div. of Refrig. Eng. Co. Ltd.
26 Great South Road, Otahahu
P.O. Box 12072
Auckland,
New Zealand
NZ (09) 276-8524

COMPILERS: 1.2/1.3 Pascal-1
MACHINES: PDP 11/XX /23
OPERATING SYS.: RSM-11M
APPLICATIONS: Real-time process control
SYSTEMS: Real time process control

Ms. Tellern Asiado
Institute of Advanced Computer Technology (I/ACT)
SGV Development Center
105 De la Rosa St.
Legaspi Village,
Makati
Phillipines
PH

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Herje Wikegard
SERN
DATOR KONSULT AB
Box 14070
Goteborg,
Sweden
SW +46-31-830800

COMPILERS: 1.3, 2.0/2.1
MACHINES: PDP 11/XX, VAX, MC68000
OPERATING SYS.: RT-11, RSX-11M
APPLICATIONS: Scientific: Tech. Dev. Proj. as Consultants, Other: Real time sys.
SYSTEMS: In house sys. or the customers own sys.

Mr. W.R. Mitchell
Chief Civil Engineer
THE HYDRO-ELECTRIC COMMISSION
4-16 Elizabeth Street
Hobart,
Tasmania
TA 30-1101

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Arthur Brown
10709 Weymouth St.
Garrett Park,
MD 20896
UE

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Paul Bauer
Advanced Control Systems
13809 Industrial Park Blvd.
Minneapolis,
MN 55441
UE

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Bob Sanford
Digital Equipment
New Jersey Turnpike Authority
c/o New Jersey Turnpike Authority
Administration Bldg.
P.O. Box 1121
New Brunswick,
NJ 08903
UE (201) 247-0900 Ext 233

COMPILERS: 2.0/2.1
MACHINES: PDP 11/XX 11/60, 11/40 & 11/04
OPERATING SYS.: RSX-11M
APPLICATIONS: Automatic Traffic Surveillance & Control
SYSTEMS: Real time

Mr. Gary Ware
EECO Incorporated
1601 East Chestnut Ave.
Santa Ana,
CA 92701
UW (714) 835-6000

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Chuck Campbell
Diagnostic Engineering
Digital Equipment Corp.
301 Rockrimmon Blvd. South
Colorado Springs,
CO 80963
UW

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Rev. James Keene
Dir. Computer Center
Saint Louis University High School
Backer Memorial
4970 Oakland Avenue
St.Louis,
MO 63110
UW

COMPILERS: 2.0/2.1
MACHINES: PDP 11/XX PDP 11/23
OPERATING SYS.: RT-11, & TSX-PLUS
APPLICATIONS: Academic: Teaching & In School Appl.
SYSTEMS: Academic

Mr. Phillip Ricker
Engineering Test Manager
Intel Corporation
Components Division
3585 S.W. 198th
M/S F4-2-546
Aloha,
OR 97007
UW (503) 642-6592

COMPILERS: 1.2/1.3
MACHINES: PDP 11/XX
OPERATING SYS.: RT-11
APPLICATIONS: Semi-conclution Testing, Cont. Sys. Proc.
SYSTEMS: In-House

Mr. John Hallick
Manager Partner
Miller Lucas Company
Systems & Programming
4811 N. Sterling Ave.
Peoria
IL 61615
UM (309) 692-5640

COMPILERS:
MACHINES:
OPERATING SYS.:
APPLICATIONS:
SYSTEMS:

Mr. Pat D'Andrea
Sr. Software Engineer
MCC Powers
2942 MacArthur Blvd.
Northbrook,
IL 60062
UW (312) 272-9555

COMPILERS: 2.0/2.1
MACHINES: PDP 11/XX, MC68000
OPERATING SYS.: RSX-11M
APPLICATIONS: Energy Mgmt/ Temperature Controls
SYSTEMS:

Mr. Steven Brecher
Software Supply
4618 E. 6th Street
Long Beach,
CA 90814
UW (213) 434-3723

COMPILERS: 1.2/1.3, 2.0/2.1
MACHINES: PDP 11/XX
OPERATING SYS.: RT-11 4.0, RSX-11M 4.0
APPLICATIONS: Support of Oregon Software Customers
SYSTEMS: OEM

Mr. Ronald Webster
Computer Services
Arizona State University
ECA 108
Tempe,
AZ 85287
UW (602) 965-1203

COMPILERS: 2.0/2.1
MACHINES: PDP 11/XX
OPERATING SYS.: RSTS/E
APPLICATIONS: Acedemic
SYSTEMS: Acedemic

Mr. Glenn Stearns
Software Development Manager
Microvertics Corp.
1394 Shorebird Way
Mountain View,
CA 94043
UW (415) 961-9454

COMPILERS: 1.2/1.3, 2.0/2.1
MACHINES: PDP 11/XX
OPERATING SYS.: RT-11 4.00B, RSX-11 4.0
APPLICATIONS: Bus. Mgmt. Sys.
SYSTEMS: Sell to end user

Mr. Peter Schmitz
Software Engineer
GenRad S.P.D.
4620 N. 16th Street
Phoenix,
AZ 85016
UW (602) 264-2475

COMPILERS: 1.2
MACHINES: PDP 11/XX
OPERATING SYS.: RT-11
APPLICATIONS: Scientific
SYSTEMS: OEM

Mr. Martin W. Sivula
System's Manager
Lunenburg Public Schools
1079 Massachusetts Ave.
Lunenburg,
MA 01462
UE (617) 582-9941 Ext 4

COMPILERS: 1.2/1.3
MACHINES: PDP 11/XX
OPERATING SYS.: RT-11, RSTS/E
APPLICATIONS: Scientific: Some, Business, Academic
SYSTEMS: Academic

Mr. Abe Getzler
Systems Analyst
Brooklyn Union Gas
195 Montague St.
Brooklyn,
NY 11201
UE (212) 403-2428

COMPILERS: 1.1
MACHINES: PDP 11/XX
OPERATING SYS.: RSX-11M, IAS
APPLICATIONS: Mobile Dispatching
SYSTEMS: Motorola

Mr. John Norman
Academic Computing
Grinnell College
Grinnell,
IA 50112
UM (515) 236-2570

COMPILERS: 1.2/1.3, 2.0/2.1
MACHINES: PDP 11/XX 11/70
OPERATING SYS.: RSTS/E
APPLICATIONS: Coursework, Computer assisted instruction
SYSTEMS: Academic

Mr. Jerry Pitzl
Associate Professor
MACALESTER COLLEGE
1600 Grand Ave.
Saint Paul,
MN 55105
UM (612) 696-6291

COMPILERS: 2.0/2.1
MACHINES: PDP 11/70, VAX (perhaps '83)
OPERATING SYS.: RSTS/E
APPLICATIONS: Academic
SYSTEMS: Academic

Mr. Alex Neussendorfer
Control Data Corp.
3965 Meadowbrook Rd.
St. Louis Park,
MN 55426
UM (612) 935-0366

COMPILERS: 2.1
MACHINES: PDP 11/XX, PDP 11-34, PDP 11-44
OPERATING SYS.: RSX-11M 4.0, 4.1
APPLICATIONS: Business: (Printed Circuit Board Mfg.)
SYSTEMS: Buxiness

Mr. Scott Snadow
General Dynamics
P.O. Box 2507
Mail Zone 4-68
Pomona,
CA 91769
UM (714) 620-7511 Ext 4779

COMPILERS: 2.0/2.1
MACHINES: PDP 11/XX
OPERATING SYS.: RSX-11M
APPLICATIONS: Scientific: Various, Other: Sys.-Related
SYSTEMS:

Ms. Cynthia Dunn
Applied Automation, Inc.
Pawhuska Road 205 ARB
Bartlesville,
OK 74004
UM (918) 661-1915

COMPILERS: 2.0
MACHINES: MC68000-EXORmacs
OPERATING SYS.: VERSAdos
APPLICATIONS: Scientific
SYSTEMS: EOM